Remote Shared Access To A Classroom Robotics Lab

William Harris
Dept. of CS
Medgar Evers College
Brooklyn, NY
wharris@acm.org

David Arnow
Turing's Craft, Inc.
Brooklyn, NY 11210
arnow@turingscraft.com

ABSTRACT
Robot programming is a quintessential hands-on computing activity, and this rightfully accounts for much of its growing popularity in the CS curriculum. However, as robot programming moves from an elective curiosity into the mainstream of the curriculum, this hands-on character will create logistical challenges of lab availability to students. Remote access to the lab during off-hours can ameliorate this problem.

Keywords
Robot programming, programming instruction, automated checking of programming exercises

1. INTRODUCTION
Robot programming is an exciting, relatively new addition to the computer science curriculum, one that articulates with multiple points of that curriculum, from the high school level to undergraduate senior projects. One reason for its effectiveness in capturing student interest is its concrete, hands-on character. Ironically, as robotics programming moves from the experimental section or the optional elective to the mainstream of the CS curriculum in a department, the need to satisfy the hands-on, concrete aspect of the subject will become a resource bottleneck. This paper reports work in progress to extend a widely used general purpose asynchronous learning tool, CodeLab™, to facilitate supportive remote access to a robotics lab for introductory robotics programming students.

2. BACKGROUND

The Robotics Laboratory, housed in the Major R. Owens NASA Aerospace Educational Laboratory (AEL) at Medgar Evers College of The City University of New York, is the primary site used to teach our robotics classes. Several groups of students (from elementary school through college) have experienced the Robotics Lab, and have completed sample lessons that introduce robot and robotics fundamentals. The Robotics Lab has been used to teach several robot programming languages (including: RCX Code, RoboLab, NQC, and XSLisp).

A variety of robot languages have been presented, over recent years, to a diverse set of populations at the lab. Elementary school students (6-graders) have studied RoboLab. High School students have studied RCX Code, RoboLab, and NQC. College students have used these languages as well in a Digital Systems course and have also used XSLisp at the lab in their Artificial Intelligence course.

The elementary school students were part of "RoboCamp 2005**":** an Intensive week-long LEGO Robotics Summer Camp for rising six-graders. The camp was a collaborative partnership between LEGO Education, the Education Technology Think Tank, Medgar Evers College, and the Crown School for Law and Journalism (PS 161). The middle school students were selected from the College's NASA SEMAA Program (Science, Engineering, Mathematics, and Aerospace Academy (SEMAA), and took part in an 8-week program. High school students participated in a 10-week program, and were part of the New York State Science, Technology, Engineering (STEP) Program at Medgar Evers College. The College students were MEC Computer Science Majors.

Currently, use of this off-line robotics lab is limited by the requirement of physical access, the need for the presence of instructional and supervisory personnel, and delays resulting from the learning curve of visiting students. The clear benefit that would result from providing structured online access to the lab leads us to CodeLab.

CodeLab [1] is the commercial version of an academic NSF project[2], WebToTeach [3, 4]. The pedagogy behind CodeLab mimics techniques used

widely in other subjects, such as mathematics and foreign language study. The main idea of the pedagogy is to provide large numbers of self-paced, highly interactive exercises that focus on key ideas of programming. These exercises are intended to augment, rather than to replace, the traditional "whole program" assignments in the first year of undergraduate study. CodeLab is a web-based tool that enables faculty to assign exercises to students and monitor student progress. For the student, CodeLab provides experience with fundamental elements of syntax, semantics, and basic usage of the programming language. The tool provides immediate feedback on correctness and often offers suggestions for fixing errors. Students can proceed at their own pace, subject to deadlines imposed at the instructor's discretion.

For the faculty member, CodeLab automatically checks student work for correctness. A dynamic roster is built into CodeLab to track student performance and maintain a record of submissions. CodeLab's helpful feedback and hints give students on-the-spot assistance, which reduces the number of students coming to faculty office hours with low-level questions. One instructor stated: ″I am getting more questions about concepts, software engineering and problem-solving and fewer questions about basics" [5]. Successful performance on CodeLab exercises certifies that the student has written code that correctly solves a stated problem. This provides instructors with an additional indicator that students coming out of an introductory class have obtained mastery of the topics covered by CodeLab.

From its beginning as an academic NSF project, a development goal of CodeLab was to reduce attrition. By providing a setting where students can master the syntax, semantics, and common usage of a programming language, CodeLab can help flatten the student learning curve in CS1. For example, one instructor reported a reduction of student withdrawals six weeks into the term from the usual 30 (out of 150) to only 2 [6]. CodeLab has also been used by several departments (for example, Brooklyn College, Blackburn College, the University of Alberta, and The Citadel) in courses other than CS1 (e.g. Data Structures or an introduction to OOP) to ease the introduction of a new language or provide a refresher of a previously taught language, without devoting substantial class time teaching or reviewing the language fundamentals. Students with programming experience have used CodeLab to quickly get up to speed on the new language, freeing the instructor to focus on the concepts of the course rather than on language details.

3. THE PROJECT

In order to increase access to the lab, and realize the benefits of CodeLab pedagogy in the context of robot programming we are integrating CodeLab with a robotics lab, and developing a suitable set of meaningful CodeLab robotics programming exercises to use in connection with this arrangement.

The traditional architecture of CodeLab involves multiple separate, typically remote, *testing* servers that have the responsibility for taking the student code submission, incorporating it in a code harness, compiling and executing the resulting program, and generating a response to the student. The servers are responsible for different languages, and so supporting a robotics programming language such as NQC in the context of CodeLab means constructing a suitable testing server: a process that can receive NQC code fragments, and with suitable harnesses build an executable NQC program and run it on a robot.

Typical non-robotics student programs run on a general purpose computer, and interact with the environment at most through file creation. In such cases, the CodeLab testing servers are able to ascertain correctness or discover particular flaws in the student code logic by examining the state of the executing program or its file system environment. In the case of robotics programming, correctness is more difficult to establish because it depends on an environment (robot motion) that is more complex, and harder to programmatically access and analyze.

For this reason, our current system will in some cases not provide definitive feedback on correctness, but rather capture the resulting robot motion in a short movie and make that movie available to the student user. Although this now shifts a significant part of the onus of evaluation to the student user herself, the CodeLab mechanism can still be used to provide immediate feedback on compilation errors and on logical errors when such errors are detectable in the NQC program state itself. In any case, video feedback is essential to retaining at least the sense of the hands-on concrete character that makes robotics programming so attractive in the first place.

The NQC testing server that accomplishes this is itself divided into two servers: the CodeLab front-end

which interacts with CodeLab and plays the role of a testing server and the robot back-end, which directly manages the robot and the camera.

The use scenario then is as follows. A student submits a solution to a robot programming exercise to CodeLab. As is true of most CodeLab exercises, the student's submission is not an entire program but a snippet of code, concentrating on one aspect of NQC or a particular robot programming technique. An NQC cross compiler is used on the front-end testing server to check the syntax, and offer some semantic checks as well, before executing them on the robot. When the program checks out, the front-end testing server, taking on the role of a client, contacts the testing back-end server in the networked robotics lab, and submits the compiled NQC program, along with various identifying information. The back-end server loads the program into a robot and lets the robot execute the program. A video camera then records the action, and the back-end server saves the short video file, returns the status and video access information to the front-end server. With that information in hand, the front-end server can continue in the typical role of any CodeLab testing server, passing the status and access information back to the main CodeLab server which makes the video available to the student.

4. REFERENCES

[1] Turing's Craft – The Exercises, http://www.turingscraft.com/exers.php

[2] Arnow, D. & Weiss, G. An Asynchronous Learning Network Tool for Improving CS Education and Retention Rates, Proposal to the National Science Foundation, EHR-DUE CCLI-EMD Program.

[3] WebToTeach: A Web-based Automated Program Checker, Frontiers in Education (FIE99), San Juan, Puerto Rico, November, 1999. (With Oleg Barshay).

[4] WebToTest: On-line Programming Examinations Using WebToTeach, ITiCSE 99, Cracow, Poland, June, 1999. (With Oleg Barshay).

[5] Rose Williams, SUNY at Binghamton, private communication, 2003.